Maxent Models and Discriminative Estimation

Christopher Manning
CS224N/Ling280

Introduction
- So far we’ve looked at “generative models”
  - Language models, Naive Bayes, IBM MT
- In recent years there has been extensive use of conditional or discriminative probabilistic models in NLP, IR, Speech (and ML generally)
  - Because:
    - They give high accuracy performance
    - They make it easy to incorporate lots of linguistically important features
    - They allow automatic building of language independent, retargetable NLP modules

Joint vs. Conditional Models
- Joint (generative) models place probabilities over both observed data and the hidden stuff (generate the observed data from hidden stuff):
  - All the best known StatNLP models: \( P(c, d) \)
    - \( n \)-gram models, Naive Bayes classifiers, Hidden Markov models, probabilistic context-free grammars
- Discriminative (conditional) models take the data as given, and put a probability over hidden structure given the data:
  - Logistic regression, conditional loglinear models, maximum entropy Markov models, conditional random fields, (SVMs, perceptrons)

Bayes Net/Graphical Models
- Bayes net diagrams draw circles for random variables, and lines for direct dependencies
- Some variables are observed; some are hidden
- Each node is a little classifier (conditional probability table) based on incoming arcs

Conditional models work well: Word Sense Disambiguation
- Even with exactly the same features, changing from joint to conditional estimation increases performance
- That is, we use the same smoothing, and the same word-class features, we just change the numbers (parameters)

Features
- In these slides and most maxent work: features are elementary pieces of evidence that link aspects of what we observe \( d \) with a category \( c \) that we want to predict.
- A feature has a (bounded) real value: \( f: C \times D \rightarrow \mathbb{R} \)
- Usually features specify an indicator function of properties of the input and a particular class (every one we present is). They pick out a subset.
  - \( f(c, d) = \Phi(d) \land c = c_i \) [Value is 0 or 1]
- We will freely say that \( \Phi(d) \) is a feature of the data \( d \), when, for each \( c_i \), the conjunction \( \Phi(d) \land c = c_i \) is a feature of the data-class pair \( (c, d) \).
Features

- For example:
  \[ f_1(c, d) = \begin{cases} \text{NN} \land \text{islower}(w_0) \land \text{ends}(w_0, \text{"d"}) \end{cases} \]
  \[ f_2(c, d) = \begin{cases} \text{NN} \land w_{-1} = \text{"to"} \land t_{-1} = \text{"TO"} \end{cases} \]
  \[ f_3(c, d) = \begin{cases} \text{VB} \land \text{islower}(w_0) \end{cases} \]

- Models will assign each feature a \textbf{weight}
- Empirical count (expectation) of a feature:
- Model expectation of a feature:

Example: Text Categorization

(Zhang and Oles 2001)

- Features are a \textit{word} in document and \textit{class} (they do feature selection to use reliable indicators)
- Tests on classic Reuters data set (and others)
  - Naïve Bayes: 77.0% \( F_1 \)
  - Linear regression: 86.0%
  - Logistic regression: 86.4%
- Support vector machine: 86.5%
- Emphasizes the importance of \textit{regularization} (smoothing) for successful use of discriminative methods (not used in most early NLP/IR work)

Other Maxent Examples

- Sentence boundary detection (Mikheev 2000)
  - Is period end of sentence or abbreviation?
- PP attachment (Ratnaparkhi 1998)
  - Features of head noun, preposition, etc.
- Language models (Rosenfeld 1996)
  - \( P(w_0 | w_{-m}, \ldots, w_1) \). Features are word n-gram features, and trigger features which model repetitions of the same word.
- Parsing (Ratnaparkhi 1997; Johnson et al. 1999, etc.)
  - Either: Local classifications decide parser actions or feature counts choose a parse.

Feature-Based Models

- The decision about a data point is based only on the \textit{features} active at that point.

Example: POS Tagging

- Features can include:
  - Current, previous, next words in isolation or together.
  - Previous (or next) one, two, three tags.
  - Word-internal features: word types, suffixes, dashes, etc.

Conditional vs. Joint Likelihood

- We have some data \((d, c)\) and we want to place probability distributions over it.
- A \textit{joint} model gives probabilities \( P(d, c) \) and tries to maximize this likelihood.
  - It turns out to be trivial to choose weights: just relative frequencies.
- A \textit{conditional} model gives probabilities \( P(c|d) \). It takes the data as given and models only the conditional probability of the class.
  - We seek to maximize conditional likelihood.
  - Harder to do (as we’ll see…)
  - More closely related to classification error.
Feature-Based Classifiers

- Linear classifiers:
  - Classify from features sets \( f_i \) to classes \( c \).
  - Assign a weight \( \lambda \) to each feature \( f_i \).
  - For a pair \((c, d)\), features vote with their weights:
    \[
    \text{vote}(c) = \sum \lambda f_i(c, d)
    \]
  - Choose the class \( c \) which maximizes \( \sum \lambda f_i(c, d) \) – VB
  - There are many ways to choose weights
    - Perceptron: find a currently misclassified example, and nudge weights in the direction of a correct classification.

Quiz question!

- Assuming exactly the same set up (2 class decision: NN or VB; 3 features defined as before, maxent model), what are:
  - \( P(c = "NN" | w_1 = "the", w_2 = "aid") \)
  - \( P(c = "VB" | w_1 = "the", w_2 = "aid") \)
  - \( 1.2 f(c, d) = [c = "NN" A islower(w_2) A ends(w_2, "d")]) \)
  - \( -1.8 f(c, d) = [c = "NN" A w_2 = "to" A f(c, d) = "to"] \)
  - \( 0.3 f(c, d) = [c = "VB" A islower(w_3)] \)

\[
P(c|d, \lambda) = \frac{\exp \left( \sum \lambda f_i(c, d) \right)}{\sum \exp \left( \sum \lambda f_i(c', d) \right)}
\]

Comparison to Naive-Bayes

- Naive-Bayes is another tool for classification:
  - We have a bunch of random variables (data features) which we would like to use to predict another variable (the class):
    \[
P(c|d, \lambda) = \frac{P(c) \prod P(f_i|c)}{\prod P(f_i|c')}
    \]

- The Naive-Bayes likelihood over classes is:
  \[
P(c|d, \lambda) = \frac{\exp \left( \log P(c) + \sum \log P(f_i|c) \right)}{\sum \exp \left( \log P(c') + \sum \log P(f_i|c') \right)}
  \]

Comparison to Naive-Bayes

- The primary differences between Naive-Bayes and maxent models are:
  - Naive-Bayes
    - Trained to maximize joint likelihood of data and classes.
    - Features assumed to supply independent evidence.
    - Features weights can be set independently.
    - Features must be of the conjunctive form.
  - Maxent
    - Trained to maximize the conditional likelihood of classes.
    - Features weights take feature dependence into account.
    - Feature weights must be mutually estimated.
    - Features need not be of the conjunctive form (but usually are).
Example: Sensors

**Reality**
- Raining
- Sunny

**NB Model**
- **NB FACTORS:**
  - P(s) = 3/8
  - P(r,+,+) = (3/4)(3/4)
  - P(s,+,+) = (3/4)(3/4)
  - P(+|r) = 3/4
  - P(+|s) = 1/4

**PREDICTIONS:**
- P(+,+,r) = 3/8
- P(−,+,r) = 1/8
- P(+,+,s) = 1/8
- P(−,+,s) = 1/8
- P(−,−,r) = 1/8
- P(−,−,s) = 6/7

Example: Stoplights

**Reality**
- Lights Working
- Lights Broken

**NB Model**
- **NB FACTORS:**
  - P(w) = 6/7
  - P(r|w) = 1/2
  - P(r|b) = 1
  - P(g|w) = 1/2
  - P(g|b) = 0

**NB FACTORS:**
- P(s) = 3/8
- P(r,+,+) = (3/4)(3/4)
- P(s,+,+) = (3/4)(3/4)
- P(+|r) = 3/4
- P(+|s) = 1/4

**PREDICTIONS:**
- P(+,+,r) = 3/8
- P(−,+,r) = 1/8
- P(+,+,s) = 1/8
- P(−,+,s) = 1/8
- P(−,−,r) = 1/8
- P(−,−,s) = 6/7

Exponential Model Likelihood

- **Maximum Likelihood (Conditional) Models:**
  - Given a model form, choose values of parameters to maximize the (conditional) likelihood of the data.

- **Exponential model form,** for a data set (C,D):

\[
\log P(C|D,\lambda) = \sum_{i=1}^{M} \log P(c_i|d_i,\lambda) = \sum_{i=1}^{M} \log \sum_{j=1}^{L} \exp \lambda_j f(c_i,d_i) / \sum_{j=1}^{L} \exp \lambda_j f(c_i,d_i)
\]

Building a Maxent Model

- **Define features (indicator functions) over data points.**
  - Features represent sets of data points which are distinctive enough to deserve model parameters.
  - Words, but also "word contains number", "word ends with img" errors.

- For any given feature weights, we want to be able to calculate:
  - Data (conditional) likelihood
  - Derivative of the likelihood wrt each feature weight
  - Use expectations of each feature according to the model

- Find the optimum feature weights (next part).
The Likelihood Value

- The (log) conditional likelihood is a function of the iid data \( (C,D) \) and the parameters \( \lambda \):
  \[
  \log P(C|D,\lambda) = \log \prod_{i \in [1,m]} P(c_i|d,\lambda) = \sum_{i \in [1,m]} \log P(c_i|d,\lambda)
  \]
- If there aren’t many values of \( c \), it’s easy to calculate:
  \[
  \log P(C|D,\lambda) = \sum_{i \in [1,m]} \log \exp \sum_{d \in [1,n_c]} \lambda_d f(c,d)
  \]
- We can separate this into two components:
  \[
  \log P(C|D,\lambda) = \sum_{i \in [1,m]} \log \exp \sum_{d \in [1,n_c]} \lambda_d f(c,d)
  \]
- The log-likelihood is a function of the iid data

The Derivative I: Numerator

\[
\frac{\partial N(\lambda)}{\partial \lambda} = \frac{\partial}{\partial \lambda} \sum_{(c,d) \in \mathcal{D}} \log \exp \sum_{d \in [1,n_c]} \lambda_d f(c,d) = \frac{\partial}{\partial \lambda} \sum_{d \in [1,n_c]} \sum_{(c,d) \in \mathcal{D}} \lambda_d f(c,d)
\]

Derivative of the numerator is: the empirical count\((f_c, c)\)

The Derivative II: Denominator

\[
\frac{\partial M(\lambda)}{\partial \lambda} = \frac{\partial}{\partial \lambda} \sum_{(c,d) \in \mathcal{D}} \log \sum_{d \in [1,n_c]} \lambda_d f(c,d)
\]

The Derivative III

\[
\frac{\partial \log P(C|D,\lambda)}{\partial \lambda} = \text{actual count}(f_c, C) - \text{predicted count}(f_c, \lambda)
\]

- The optimum parameters are the ones for which each feature’s predicted expectation equals its empirical expectation. The optimum distribution is:
  - Always unique (but parameters may not be unique)
  - Always exists (if feature counts are from actual data).
- These models are also called maximum entropy models because we find the model having maximum entropy and satisfying the constraints:
  \[
  E_p(f_j) = E_p(f_j), \forall j
  \]

Fitting the Model

- To find the parameters \( \lambda_1, \lambda_2, \lambda_3 \)
- write out the conditional log-likelihood of the training data and maximize it
- The log-likelihood is concave and has a single maximum; use your favorite numerical optimization package
- Good large scale techniques: conjugate gradient or limited memory quasi-Newton

Fitting the Model Generalized Iterative Scaling

- A simple optimization algorithm which works when the features are non-negative
- We need to define a slack feature to make the features sum to a constant over all considered pairs from \( D \times C \)
- Define \( M = \max_{d \in D} \sum_{c \in C} f_{d,c} \)
- Add new feature
  \[
  f_{new}(d,c) = M - \sum_{j \in C} f_{d,j}
  \]
**Generalized Iterative Scaling**
- Compute empirical expectation for all features \( E_i(f_j) = \frac{1}{N} \sum (d,c) f_i(d,c) \)
- Initialize \( \lambda_j = 0, j = 1...m+1 \)
- Repeat
  - Compute feature expectations according to current model \( E_i(f_j) = \frac{1}{N} \sum (d,c) P(d,c) f_i(d,c) \)
  - Update parameters \( \lambda_j = \lambda_j + \frac{1}{M} \log \left( \frac{E_i(f_j)}{E_i(f_j)} \right) \)
- Until converged

**Maximum Entropy Models**
- An equivalent approach:
  - Lots of distributions out there, most of them very spiked, specific, overfit.
  - We want a distribution which is uniform except in specific ways we require.
  - Uniformity means high entropy – we can search for distributions which have properties we desire, but also have high entropy.

**(Maximum) Entropy**
- Entropy: the uncertainty of a distribution.
- Quantifying uncertainty (“surprise”):
  - Event \( x \)
  - Probability \( p_x \)
  - “Surprise” \( \log(1/p_x) \)
- Entropy: expected surprise (over \( p \)):
  \[
  H(p) = E_x \log \left( \frac{1}{p_x} \right) \\
  H(p) = -\sum_x p_x \log p_x 
  \]

**Maxent Examples I**
- What do we want from a distribution?
  - Minimize commitment = maximize entropy.
  - Resemble some reference distribution (data).
- Solution: maximize entropy \( H \), subject to feature-based constraints:
  \[
  E_x[f_i] = E_p[f_i] \iff \sum_x p_x = C_i 
  \]
- Adding constraints (features):
  - Lowers maximum entropy
  - Raises maximum likelihood of data
  - Brings the distribution further from uniform
  - Brings the distribution closer to data

**Maxent Examples II**
- Lets say we have the following event space:
  \[
  \begin{align*}
  NN & 0.5 \\
  NNS & 0.5 \\
  NNP & 0.5 \\
  NNPS & 0.5 \\
  VBZ & 0.5 \\
  VBD & 0.5 
  \end{align*}
  \]
  - and the following empirical data:
  \[
  \begin{align*}
  NN & 3 \\
  NNS & 5 \\
  NNP & 11 \\
  NNPS & 13 \\
  VBZ & 3 \\
  VBD & 1 
  \end{align*}
  \]
  - Maximize \( H \):
  \[
  \frac{1}{e} \frac{1}{e} \frac{1}{e} \frac{1}{e} \frac{1}{e} \frac{1}{e} 
  \]
  - ... want probabilities: \( E[NN,NNS,NNP,NNPS,VBZ,VBD] = 1 \)

**Maxent Examples III**
- Ignorance is preferable to error and he is less remote from the truth who believes nothing than he who believes what is wrong – Thomas Jefferson (1781)
Maxent Examples IV

- Too uniform!
- N* are more common than V*, so we add the feature \( f_N = \{NN, NNS, NNP, NNPS\} \), with \( E[f_N] = 32/36 \)
- ... and proper nouns are more frequent than common nouns, so we add \( f_P = \{NNP, NNPS\} \), with \( E[f_P] = 24/36 \)
- ... we could keep refining the models, e.g. by adding a feature to distinguish singular vs. plural nouns, or verb types.

Convexity

\[ f(\sum_i w_i x_i) \geq \sum_i w_i f(x_i) \sum_i w_i = 1 \]

Convexity guarantees a single, global maximum because any higher points are greedily reachable.

Convexity II

- Constrained \( H(p) = -\sum x \log x \) is convex:
  - \( -x \log x \) is convex
  - \( -\sum x \log x \) is convex (sum of convex functions is convex).
  - The feasible region of constrained \( H \) is a linear subspace (which is convex)
  - The constrained entropy surface is therefore convex.

Feature Overlap

Maxent models handle overlapping features well.
Unlike a NB model, there is no double counting!

Example: NER Overlap

Grace is correlated with PERSON, but does not add much evidence on top of already knowing prefix features.

Feature Weights

<table>
<thead>
<tr>
<th>Feature Type</th>
<th>Feature</th>
<th>PERS</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous word</td>
<td>at</td>
<td>0.73</td>
<td>0.94</td>
</tr>
<tr>
<td>Current word</td>
<td>at</td>
<td>0.63</td>
<td>0.03</td>
</tr>
<tr>
<td>Beginning bigram</td>
<td>at</td>
<td>0.45</td>
<td>0.04</td>
</tr>
<tr>
<td>Current POS tag</td>
<td>NNP</td>
<td>0.47</td>
<td>0.45</td>
</tr>
<tr>
<td>Prev and cur tags</td>
<td>IN NNP</td>
<td>0.10</td>
<td>0.14</td>
</tr>
<tr>
<td>Previous state</td>
<td>Other</td>
<td>0.70</td>
<td>0.92</td>
</tr>
<tr>
<td>Current signature</td>
<td>Xx</td>
<td>0.80</td>
<td>0.40</td>
</tr>
<tr>
<td>Prev state, cur sig</td>
<td>O-x</td>
<td>0.68</td>
<td>0.37</td>
</tr>
<tr>
<td>Prev-cur next sig</td>
<td>x-x-x</td>
<td>0.68</td>
<td>0.37</td>
</tr>
<tr>
<td>P. state - p-cur sig</td>
<td>O-x-x</td>
<td>0.20</td>
<td>0.82</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>-0.58</td>
<td>2.68</td>
</tr>
</tbody>
</table>

Local Context

<table>
<thead>
<tr>
<th>Prev</th>
<th>Cur</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>Other</td>
<td>at</td>
</tr>
<tr>
<td>Word</td>
<td>at</td>
<td>Grace</td>
</tr>
<tr>
<td>Tag</td>
<td>IN</td>
<td>NNP</td>
</tr>
<tr>
<td>Sig</td>
<td>x</td>
<td>Xx</td>
</tr>
</tbody>
</table>

Feature Interaction

Maxent models handle overlapping features well, but do not automatically model feature interactions.
Feature Interaction

- If you want interaction terms, you have to add them:

  **Empirical**

  ![Feature Interaction Diagram]

  A disjunctive feature would also have done it (alone):

  ![Feature Interaction Diagram]

**Example: NER Interaction**

Previous-state and current-signature have interactions, e.g. P=PERS>C=Xx indicates C=PERS much more strongly than C=Xx and P=PERS independently.

This feature type allows the model to capture this interaction.

**Local Context**

<table>
<thead>
<tr>
<th>Feature Type</th>
<th>Feature</th>
<th>PERS</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous word</td>
<td>at</td>
<td>0.73</td>
<td>0.94</td>
</tr>
<tr>
<td>Current word</td>
<td>Grass</td>
<td>0.03</td>
<td>0.06</td>
</tr>
<tr>
<td>Beginning bigram</td>
<td>G</td>
<td>0.45</td>
<td>0.64</td>
</tr>
<tr>
<td>Current POS tag</td>
<td>NNP</td>
<td>0.47</td>
<td>0.45</td>
</tr>
<tr>
<td>Prev and cur sig</td>
<td>IN NN</td>
<td>0.10</td>
<td>0.14</td>
</tr>
<tr>
<td>Prev state</td>
<td>Other</td>
<td>0.70</td>
<td>0.92</td>
</tr>
<tr>
<td>Current sig</td>
<td>Xx</td>
<td>0.80</td>
<td>0.46</td>
</tr>
<tr>
<td>Prev state, cur sig</td>
<td>O-Xx</td>
<td>0.68</td>
<td>0.37</td>
</tr>
<tr>
<td>Prev cur next sig</td>
<td>Xx-Xx</td>
<td>0.69</td>
<td>0.82</td>
</tr>
<tr>
<td>P. state</td>
<td>p-cur sig</td>
<td>O-Xx</td>
<td>-0.20</td>
</tr>
<tr>
<td>Sig</td>
<td>x</td>
<td>-0.58</td>
<td>2.64</td>
</tr>
</tbody>
</table>

**Classification II**

- $D$ may be huge or infinite, but only a few $d$ occur in our data.
- What if we add one feature for each $d$ and constrain its expectation to match our empirical data?

  \[ \mathbb{V}(d) \in D \quad \mathbb{P}(d) = \hat{\mathbb{P}}(d) \]

  - Now, most entries of $P(c,d)$ will be zero.
  - We can therefore use the much easier sum:

  \[
  E(f) = \sum_{c,d \in \mathbb{R}_{C,D}} P(c,d) f(c,d) \\
  = \sum_{c,d \in \mathbb{R}_{C,D}, P(d)>0} P(c,d) f(c,d)
  \]

**Classification III**

- But if we've constrained the $D$ marginals

  \[
  \forall d \in D \quad \mathbb{V}(d) \in D \quad \mathbb{P}(d) = \hat{\mathbb{P}}(d)
  \]

  then the only thing that can vary is the conditional distributions:

  \[
  P(c,d) = P(c|d) \hat{\mathbb{P}}(d) \\
  = \hat{\mathbb{P}}(c|d)
  \]

  - This is the connection between joint and conditional maxent / exponential models.
  - Conditional models can be thought of as joint models with marginal constraints.
  - Maximizing joint likelihood and conditional likelihood of the data in this model are equivalent!